Chapter 11
Program Development and Programming Languages

Deborah Morley
Charles S. Parker
Learning Objectives (1)

• Understand the differences between structured programming, object-oriented programming (OOP), aspect-oriented programming (AOP), and adaptive software development.

• Identify and describe the activities involved in the program development life cycle (PDLC).

• Understand what constitutes good program design and list several tools that can be used by computer professionals when designing a program.

• Explain the three basic control structures and how they can be used to control program flow during execution.
Learning Objectives (2)

• Discuss some of the activities involved with debugging a program and otherwise ensuring it is designed and written properly.

• List some tools that can be used to speed up or otherwise facilitate program development.

• Describe several programming languages in use today and explain their key features.
Overview

• This chapter covers:
  – The most common approaches to program design and development
  – The phases of the program development life cycle (PDLC)
  – Tools that can be used to design and develop a program
  – Good program design techniques and types of program errors
  – Popular programming languages
  – Markup and scripting languages
Approaches to Program Design and Development

• **Procedural programming** separates a program into small modules that are called by the main program or another module when needed
  – Allows each procedure to be performed as many times as needed; multiple copies of code not needed
  – Prior to procedural programming, programs were one large set of instructions (used GOTO statements)

• Structured programming goes even further, breaking the program into small modules and prohibiting GOTO

• **Variables** are named memory locations that are defined for a program
  – Used to store the current value of data items used in the program
Examples of Structured Programming

Modules are arranged hierarchically in a top-down fashion, as illustrated here for a payroll application.

- Control program
  - The control program calls each module as needed, such as when it is time to compute the deductions.
  - Input
    - Each module then calls additional modules, as needed, such as to compute federal taxes.
      - Compute federal taxes
      - Compute state taxes
      - Compute retirement contribution
      - Compute other deductions
    - Gross Pay
    - Deductions
    - Net Pay
    - Output
      - Issue checks
      - Print payroll report
Object-Oriented Programming (OOP)

• **Object-oriented programming (OOP)** consist of a collection of objects that contain data and methods to be used with that data
  
  – Class: A group of objects that share some common properties
  
  – Attributes: Data that describes the object
  
  – Methods: Perform actions on an object
    
    • Can be used with different types of objects
  
  – Instance: An individual object in a class
    
    • Inherits the attributes and methods of the class
    
    • The values of attributes may vary from instance to instance
  
  – Objects can be accessed by multiple programs
Example of a Class Diagram

**FIGURE 11-2**
Button class.
This class diagram illustrates that each object (instance) in the Button class has four attributes to hold data about the current state of the button and three methods to perform actions when messages are received.

- **Attributes**
  - ButtonColor
  - ButtonSize
  - DisplayCoordinates
  - ButtonText

- **Methods**
  - Display
  - Hide
  - Dim

© 2017 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part, except for use as permitted in a license distributed with a certain product or service or otherwise on a password-protected website for classroom use.
Aspect-Oriented Programming and Adaptive/Agile Software Development

- **Aspect-oriented programming (AOP)** separates functions so program components can be developed and modified individually from one another
  - The components can be easily reused
- Adaptive software development adapts programs as they are being written
  - Typically iterative and/or incremental development
  - **Agile software development (ASD)** can create software quickly
    - Focuses on building small functional program pieces as the project progresses
    - Emphasizes teams of people working closely together (programmers, managers, business experts, customers, and so forth)
    - Some mobile developers are using continuous mobile innovation
The Program Development Life Cycle (PDLC)

• **Program development** (application software development) is the process of creating application programs

• **Program development life cycle (PDLC)** includes the five phases of program development
  – Typically takes place during the system acquisition phase of the SDLC
  – One completed, the SDLC continues
The program development life cycle (PDLC). Each phase of the program development life cycle produces some type of documentation to pass on to the next phase.
Problem Analysis

• During problem analysis, the problem is considered and the program specifications are developed
  – Specifications developed during the SDLC are reviewed by the systems analyst and the programmer (the person who will code the program)
  – Goal is to understand the functions the software must perform, determine the appropriate programming language, determine how the program interacts with other programs, etc.

• Documentation: Program specifications that outline what the program must do
Program Design

• During **program design**, the program specifications are used to develop an algorithm (the steps the program must do) for that program
  
  – Careful planning and design of a computer program are extremely important
  
  – Good program design helps the development process as well as makes revisions easier to do in the future
  
  – The coding stage cannot occur until the program design is complete and tested
Program Design Tools: Structure Charts and Flowcharts

- **Structure charts** (hierarchy charts) depict the overall organization of a program
  - Shows program modules and hierarchy

- **Flowcharts** show the step-by-step logic for a program, module, or method
  - Use special symbols and relational operators
  - Can be drawn by hand or with flowcharting software
Program Design Tools: Wireframes

- A **wireframe** is a visual representation of the overall design and logic of an application.
- Traditionally focus is on the visual functional elements.
- Now also describes the flow of logic particularly in mobile apps.
- Wireframe software can be used.

*FIGURE 11-5 Wireframes.*
Program Design Tools: Pseudocode

• **Pseudocode** uses English-like statements to outline the logic of a program rather than the flowchart’s graphical symbols

• No formal standard

---

**FIGURE 11-6**

Pseudocode. For the flowchart logic shown in Figure 11-4.

Start

counter = 0

Read a record

DO WHILE there are records to process

  IF computer_experience
    IF company_service ≥ 5 years
      Print employee_name
      Increment counter
    ELSE
      Next statement
      END IF
  ELSE
    Next statement
    END IF

ELSE

  Read another record
END DO

Print counter

Stop

© 2017 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part, except for use as permitted in a license distributed with a certain product or service or otherwise on a password-protected website for classroom use.
Program Design Tools: Unified Modeling Language (UML)

- **Unified Modeling Language (UML)** is a set of standard notations for creating business models
  - Widely used in object-oriented programs
  - Includes class diagrams and use case diagrams

**FIGURE 11-7**
Class diagrams.
This example shows one class and two instances of that class.
Control Structures

• A control structure illustrates when, how, and in what order the statements in a computer program, module, or method are performed

• **Sequence control structure**
  – Series of statements that follow one another

• **Selection control structure**
  – Multiple paths, direction depends on the result of a certain condition
    • If-then-else (two possibilities)
    • Case control structure (more than two possibilities)
Control Structures (cont’d)

• **Repetition control structure** (iteration control structure)
  – Series of statements in a loop that are repeated until a particular condition is met
    • Do while structure
      – Statements are repeated as long as a condition is true
      – If initially false, statements will not be executed
    • Do until structure
      – Statements are repeated until a condition is true
      – Statements are always executed one time
Examples of Control Structures

FIGURE 11-8
The three fundamental control structures. Note that each structure example has only one entry point and only one exit point.
Good Program Design

• Good program design
  – Be specific
    • All things the program must do or consider must be specified
  – Follow the one-entry-point/one-exit-point rule
    • Only one entry and one exit for each control structure
  – No infinite loops or logic errors
    • An infinite loop is a series of steps that repeat forever
    • These and other logic errors need to be uncovered during testing
Example of Writing Instructions for a Computer vs. a Person

**DIRECTIONS FOR PERSON**

1. Please make a piece of toast with margarine for me.

**DIRECTIONS FOR COMPUTER**

1. Take one slice of bread out of the bag of bread, then close the bag.
2. Put the bread into the slot on the toaster, narrow edge first with the widest part of the bread fitting into the widest part of the slot.
3. Push the start lever on the toaster down.
4. When the toast pops up, remove the toast from the toaster.
5. Place the toast on a plate.
6. Open the silverware drawer, take out a knife, then close the silverware drawer.
7. Open the refrigerator, remove the margarine, then shut the refrigerator.
8. Take the lid off the margarine.
9. Scoop out one teaspoon of margarine with the knife.
10. Spread the margarine on the top side of the toast, evenly covering that surface of the toast.
11. Place the lid back on the margarine.
12. Open the refrigerator, replace the margarine, then shut the refrigerator.

**FIGURE 11-9**

Writing instructions for a computer versus a person. A computer requires step-by-step instructions.
Program Design Testing

• Program design testing
  – Design should be tested to ensure logic is correct
    • Desk check flowcharts on paper
    • Tracing tables can help keep track of variable values

• Documentation: Design specifications
  • Design specifications; often using design tools like structure charts, flowcharts, wireframes, pseudocode, UML models, revised UI designs, etc.
  • Include any test data and results from desk checking
Example of Desk Checking a Flowchart

FIGURE 11-10
Desk checking a flowchart.
Program Coding

• **Program coding** involves writing code using a programming language
  – Choosing a programming language
    • Suitability to the application
    • Integration with other programs
    • Standards for the company
    • Programmer availability
    • Portability if being run on multiple platforms
    • Development speed
Coding Process and Standards

• Programs are coded in a specific language using the appropriate program language application
  – The resulting program is **source code**

• Coding standards are rules set by organizations designed to standardize programming styles
  – Makes programs more readable and easier to maintain
  – Includes the proper use of comments to:
    • Identify the programmer and last modification date
    • Explain variables used in the program
    • Identify the main parts of the program
Examples of Program Comments

**COMMENTS**

Comments are usually preceded by a specific symbol (such as `*`, `C`, `#`, or `/**`); the symbol used depends on the programming language being used. Comment lines are ignored by the computer.

Comments at the top of a program should identify the name and author of the program, date written and last modified, purpose of the program, and variables used in the program.

Comments in the main part of a program should indicate what each section of the program is doing. Blank comment lines can also be used to space out the lines of code, as needed for readability.

Lines not marked as comments are executable code.

```
* This program inputs two numbers, computes their sum, and displays the sum.
* Written by: Deborah Morley 3/12/16

* Variable list
  * SUM: Running sum
  * CNTR: Counter
  * NUM: Number inputted

  REAL SUM, CNTR, NUM

* INITIALIZE VARIABLES
  SUM = 0
  CNTR = 0

* INPUT NUMBER, ADD IT TO THE SUM, INCREMENT COUNTER, AND THEN
  * REPEAT UNTIL TWO NUMBERS HAVE BEEN ENTERED

  DO 10 CNTR = 1, 2
```

**FIGURE 11-11**

Program comments.
Reusable Code

• Reusable code is pretested, error-free code segments that can be used over and over again with minor modifications
  – Can greatly reduce development time
  – Class libraries

• Documentation: Documented source code
  – Completed source code in the desired language that implements the logic illustrated by the program design specifications
  – Should include enough comments (internal documentation) so that the source code is easy to understand and update
Programming Contests

• One example is the TopCoder Open
  – Six competitions
  – Initial qualifying rounds are online
  – 12 semifinalists compete on site
  – $300,000 in prizes
• Other competitions are available online
Program Debugging and Testing

- **Program debugging and testing** is the process of ensuring a program is free of errors (bugs) and works as it is supposed to.
- Translating coded programs into executable code
  - Coded programs need to be translated from source code written by the programmer to **object code** the computer can execute.
  - Converted using a **language translator** (a program that converts source code to object code).
    - Typically included in the programming application.
Compilers, Interpreters, and Assemblers

- **Compiler**
  - Language translator that converts an entire program into machine language before executing it
  - Designed for specific programming languages such as Java or Python

- **Interpreter**
  - Translates one line of code at one time

- **Assembler**
  - Converts assembly language programs into machine language

**FIGURE 11-12**
Compiler and linkage editor. A compiler and a linkage editor convert source code into executable code.

© 2017 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part, except for use as permitted in a license distributed with a certain product or service or otherwise on a password-protected website for classroom use.
Inside the Industry

The Original Program “Bug”

• A bug is an error that causes a program to malfunction
• First recorded instance of the term “bug” occurred in 1945
• Short circuit caused by a moth caught between two contacts in one of the computer’s relays

The dead moth that caused the temporary failure of the Mark II computer in 1945, thought to be the origin for the computer term bug, was taped into the actual log book for that computer.
Preliminary Debugging

• Compiler errors occur as a program is being compiled or interpreted
  – Often **syntax errors** – errors that occur when the programmer has not followed the rules of the programming language
    • Spell a command wrong, faulty punctuation, etc.
  – Error messages are typically displayed
• Run time errors occur when the program is running
  – Often **logic errors** – errors in the logic of the program
  – Program will run but produces incorrect results
  – Dummy print statements can help locate the error
Example of a Syntax Error

1. Clicking the Start button with the Debug option selected starts the compilation and debugging process.

2. If a compiler error is encountered, the application typically displays an error message.

3. This misspelled command is marked by a wavy underline.

4. The debugger displays an error list containing all compiler errors.

**FIGURE 11-13**
Syntax errors. Occur when the syntax (grammar rules) for a program is not followed precisely; they become obvious when compiling a program.
Example of a Logic Error and Dummy Print Statements

1. With logic errors, such as initializing a counter to the wrong number as shown here, the program will run but the output will be wrong.

2. Adding dummy print statements to display the values of key variables and key locations in the program can help to determine the error.

3. The dummy print statements, as well as the regular input and output messages belonging to the program, are displayed at the appropriate times when the program is executed.

4. The dummy print statements, which reveal that the loop is performed only once before the sum is displayed here, help the programmer locate the counter initialization error.

FIGURE 11-14
Logic errors. Are more difficult to identify; dummy print statements can help determine the error.
Testing

• Testing occurs after the preliminary debugging process to find any additional errors
  – Use good test data—data that is very similar to the actual data that will be used in the finished program
  – Test to see how the program handles nonstandard situations or possible input errors
  – Alpha tests are on site
  – Beta tests are outside tests

• Documentation: Completed program package
  – Test data, test results, finished program code, and other documentation
  – User documentation (users’ manual, etc.)
Program Implementation and Maintenance

• **Program implementation and maintenance** includes:
  – Implementation: Getting the tested program installed and up and running
  – Program maintenance: Updating software as needed so it continues to be useful

• Documentation: Amended program package
  – Program package should be updated to reflect new problems or issues that occur and what changes to the program were necessary
Quick Quiz (1)

1. Which approach to programming uses the concept of inheritance?
   a. procedural
   b. object-oriented
   c. structured

2. True or False: An infinite loop is an example of a logic error.

3. A program design tool that uses English-like statements to outline the logic of a program is known as ___________.

Answers:
1) b; 2) True; 3) pseudocode
Tools for Facilitating Program Development

• **Application lifecycle management (ALM)** tools create and manage an application during its entire lifecycle, from design through retirement
  
  – Requirements management keeps track of and managing the program requirements as they are defined and modified
  
  – Configuration management keeps track of the progress of a program development project
  
  – Issue tracking records issues such as bugs or other problems that arise during development or implementation
Example of Issue Tracking Software

**FIGURE 11-15**
Issue tracking software. Allows you to track issues during the development and life of an application, such as the game app shown here.
Application Generators and Device Development Tools

• Application generators are software program that helps programmers develop software
  – Macro: A sequence of saved actions that can be replayed when needed
    • Written in a macro programming language such as Visual Basic for Applications or recorded
  – Report generator: A tool that prepares reports to be used with a software program quickly and easily
  – User interface (UI) builder: Creates the menus, forms, and input screens used with a program or database

• Device development tools assist with developing embedded software to be used on devices, such as cars, ATM machines, and consumer devices
Examples of User Interface (UI) Builders

**FIGURE 11-16**
User interface (UI) builders.
SDKs, APIs, and IDEs

- A **software development kit (SDK)** is a set of development tools for a particular platform or programming language
  - Enable programmers to develop applications more quickly and easily
  - Android SDK, iOS SDK, etc.

- An **application program interface (API)** helps applications interface with a particular operating system, Web site, or device
  - Google’s Maps API, Google’s OpenSocial API, etc

- An **integrated development environment (IDE)** includes a collection of tools for one or more particular programming languages that have a common interface and are used to develop and test software for one or more platforms
Quick Quiz (2)

1. Which tool is includes a variety of tools (such as a UI builder and source code editor) for one or more languages?
   a. integrated development environment (IDE)
   b. application lifecycle management (ALM)
   c. application program interface (API)

2. True or False: A software development kit (SDK) is designed for a particular platform and allows programmers to develop applications quickly for that platform.

3. A(n) _________ is a sequence of saved actions that can be replayed whenever needed within the application program in which it was created.

Answers:
1) a; 2) True; 3) macro
Languages Used For Application Development

• A **programming language** is a set of rules, words, symbols, and codes used to write computer programs
  – To write a program, you need the appropriate software for the programming language being used

• **Markup languages** are most often used to create Web pages and Web applications
  – Use markup tags to identify elements and their properties

• **Scripting languages** are interpreted instead of compiled, so they are executed one command at a time and at the time they are run
  – Most often used to add dynamic content
Trend

Mobile UX

• UX (user experience) guidelines:
  – Determine the primary objectives of the application and your targeted audience
  – Keep the design simple, clean, efficient, and task-oriented
  – Make the application fast and responsive
  – Don’t require sign-in if not necessary
  – Be aware of emerging trends

It is important to create a good user experience (UX).
Low-Level Programming Languages

- **Low-level languages** (earliest programming languages)
  - **Machine language**
    - Written at a very low level, just using 1s and 0s
    - First generation of programming languages
  - **Assembly language**
    - Uses names and other symbols to replace some of the 1s and 0s in machine language
    - Second generation of programming languages
    - Programs take longer to write and maintain
Example of Machine and Assembly Language

![Example of Machine and Assembly Language](image)

**FIGURE 11-18**

Assembly and machine language.
High-Level Programming Languages

• **High-level languages:** Closer to natural languages
  – Machine-independent
  – Most are 3GLs and can be:
    • Procedural languages (Fortran, BASIC, COBOL, C, etc.)
    • Object-oriented languages (C++, C#, Python, Java, etc.)
  – Can be visual programming environments (VPEs), which allow programmers to create the interface graphically
    • Dragging and dropping objects and then defining their appearance and behavior
    • Often assist in creating code
  – Can be visual programming languages which create programs entirely using graphical elements
    • Scratch
Scratch
Fourth-Generation Languages (4GLs)

- **Fourth-generation languages (4GLs)** (very-high-level languages)
  - Even closer to natural languages and easier to work with than high-level languages
  - Declarative rather than procedural
    - You tell the computer what to do and it figures out how to do it
  - Commonly used to access databases
    - Structured query language (SQL)
  - Can result in less efficient code when compiled
Common Programming Languages: Fortran

- **Fortran**, originally called **FORTRAN** (FORmula TRANslator)
  - High-level programming language used for mathematical, scientific, and engineering applications
  - Still used today for high-performance computing tasks (weather forecasting)
  - Fortress is a similar program designed for high-performance computing
    - Takes advantage of multi-core processors and computers with multiple processors
    - An open source program but is not being updated
Comments are preceded by an asterisk or a C.

```fortran
REAL SUM, CNTR, NUM
*
* INITIALIZE VARIABLES
SUM = 0
*
* INPUT NUMBER, ADD IT TO THE SUM, AND THEN
* REPEAT UNTIL TWO NUMBERS HAVE BEEN ENTERED
DO 10 CNTR = 1, 2
   WRITE(*,*) 'Enter number'
   READ(*,*) NUM
   SUM = SUM + NUM
CONTINUE

10 *
* PRINT THE SUM
   WRITE(*,*) 'SUM IS ', SUM
*
END
```

Program statements can be numbered in order to control loops and other types of branching.
COBOL

- **COBOL** (Common Business-Oriented Language)
  - Designed for business transaction processing
  - Makes extensive use of modules
  - Strength lies in batch processing and its stability
  - Programs are lengthy and take a long time to write
  - Considered to be outdated but still a need for COBOL programs because existing business applications are still written in COBOL
  - New versions are evolving
    - Object-oriented COBOL programs
    - COBOL.NET
A Sample COBOL Program

Comments are preceded by an asterisk.

Most COBOL programs use a number of modules to break the program into manageable pieces. These submodules are called from the main control module using these statements.

Three submodules are used in this program.

© 2017 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part, except for use as permitted in a license distributed with a certain product or service or otherwise on a password-protected website for classroom use.
Pascal, BASIC, and Visual Basic

- **Pascal**
  - Named after mathematician Blaise Pascal
  - Created as a teaching tool to encourage structured programming
  - Contains a variety of control structures used to manipulate modules systematically

- **BASIC** (Beginner’s All-purpose Symbolic Instruction Code)
  - Easy-to-learn, high-level programming language that was developed to be used by beginning programmers
  - Used for interactive programs

- **Visual Basic**: Version of BASIC that uses a visual environment
C, C++, C#, Objective-C, Swift, and F#

- **C**: A powerful and flexible language used for a variety of applications
  - Much closer to assembly language than other high-level languages
- **C++**: Object-oriented version of C
- **C# (C-sharp)**: A hybrid of C and C++
  - Most often used to create Web and Windows applications
- **Objective-C**: A version of C used to write programs for Apple devices
  - Being replaced by Swift
- **F#**: Developed for .NET platform as an improvement to C#
A Sample C++ Program

```c++
#include <iostream.h>

void main ()
{
    // Declare and initialize variables
    float fSum = 0;
    float fNum;
    int iCntr = 0;

    // Input a number, add it to the sum, and repeat
    // until two numbers have been entered
    do
    {
        cout << "Enter number: "; // Prompt for input
        cin >> fNum;
        fSum = fSum + fNum;
        iCntr = iCntr + 1;
    } while(iCntr < 2);

    // Print the sum
    cout << "The sum of the numbers you entered is " << fSum;
}
```

FIGURE 11-22
The adding-two-numbers program written in C++.
Java and Dart

- **Java**: Object-oriented programming language commonly used for Web-based applications
  - Developed by Sun Microsystems and is now open source
  - Java programs are compiled into bytecode
    - Can run on any computer that includes Java Virtual Machine (Java VM)
    - Can be used to write Java applets (small programs inserted into Web pages and run by browsers)
- **Dart**: Open source, object-oriented programming language developed by Google
  - Designed to replace JavaScript in Web applications
  - Similar multiple data elements can be manipulated together
A Sample Java Program

The java.io package will handle the user input. " indicates all classes will be available.

// Input a number, add it to the sum, and repeat
// until two numbers have been entered
do
{  
    System.out.println("Enter number: ");
    inData = stdin.readLine();  // get number in character form
    iNum = Integer.parseInt( inData );  // convert inData to integer
    iSum = iSum + iNum;
    iCntr = iCntr + 1;
}  // while (iCntr < 2);

// Print the sum
System.out.println("The sum of the numbers you entered is " + iSum);

FIGURE 11-23
The adding-two-numbers program written in Java.
A Sample Dart Program

- Dart apps can run directly in Dartium
- For other browsers, much be compiled to JavaScript code
• **Python**: An open-source, dynamic, object-oriented language that can be used to develop a variety of applications
  
  – Can run on a variety of computers (Windows, Linux, UNIX, OS X, etc.) and on some mobile devices
  
  – Better code readability
  
  – Used by large organizations and some colleges, such as MIT, are using Python for some programming courses instead of more traditional languages
# Initialize variable
total = 0.0

# Input a number, add it to the total, and repeat
# until two numbers have been entered
for iteration in range(2):
    text = raw_input("Enter number: ")
    total = total + float(text)

# Print the sum
print "The sum of the numbers you entered is", total

FIGURE 11-25
The adding-two-numbers program written in Python.
Creating Apps Using the Android SDK and Eclipse

1. Create a new project
2. Modify the blank activity XML file to add needed elements, then edit the XML code as needed to finish the UI
3. Modify the Java code as needed to perform the needed actions
4. Run the app using the Android Emulator and then edit the app as needed
• Define the structure, layout, and general appearance of the content located on a Web page

• **Hypertext Markup Language (HTML):** The markup language designed for creating Web pages
  – Use HTML tags - Text-based codes embedded into a Web page’s source code
    • Indicate the location and appearance of content on that Web page
    • Some are paired tags
  – The Web browser, browser settings, and device used ultimately determine what the Web page will look like
Examples of HTML Tags

<table>
<thead>
<tr>
<th>TAG</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;html&gt;&lt;/html&gt;</td>
<td>Marks the beginning and end of an HTML document.</td>
</tr>
<tr>
<td>&lt;head&gt;&lt;/head&gt;</td>
<td>Marks the head section, which contains the page title and meta tags.</td>
</tr>
<tr>
<td>&lt;title&gt;&lt;/title&gt;</td>
<td>Marks the title of the Web page.</td>
</tr>
<tr>
<td>&lt;body&gt;&lt;/body&gt;</td>
<td>Contains all the content of the Web page, including text, hyperlinks, and images.</td>
</tr>
<tr>
<td>&lt;h1&gt;&lt;/h1&gt; to &lt;h6&gt;&lt;/h6&gt;</td>
<td>Formats headings larger or smaller than the regular (non-heading) text in the document; H1 is the largest text.</td>
</tr>
<tr>
<td>&lt;img&gt;</td>
<td>Indicates an image file to be inserted; attributes included within this tag specify the image filename, display size, alternative text, title, border, etc.</td>
</tr>
<tr>
<td>&lt;a&gt;&lt;/a&gt;</td>
<td>Defines a hyperlink using the specified URL; can include an image filename, hyperlink text, and other attributes.</td>
</tr>
<tr>
<td>&lt;b&gt;&lt;/b&gt;</td>
<td>Bolds text.</td>
</tr>
<tr>
<td>&lt;!-- --&gt;</td>
<td>Indicates a comment that won’t display when the Web page is viewed.</td>
</tr>
<tr>
<td>&lt;hr&gt;</td>
<td>Inserts a horizontal rule.</td>
</tr>
<tr>
<td>&lt;p&gt;</td>
<td>Inserts a paragraph break (starts a new paragraph).</td>
</tr>
</tbody>
</table>

FIGURE 11-26
Sample HTML tags.
XML and XHTML

- **XML (Extensible Markup Language)** is a set of rules for exchanging data over the Web
  - “Extensible” because the data contained in XML documents can be extracted when needed and used in a variety of ways
  - Identifies only the data itself, not the format of that data
  - XML tags are assigned to data and vary from one organization to the next
  - Allows easy retrieval and updating of data

- **XHTML (Extensible Hypertext Markup Language)** is a version of HTML that is based on XML
  - Supports both XHTML and XML tags
HTML5

- **HTML5** is the current version of HTML
  - Designed to replace both HTML and XHTML
  - Supports the creation of more complex and dynamic Web pages and applications
  - HTML5 pages have three main sections (declaration, head, and body)
  - Support new HTML5 tags
    - Audio and video tags
    - Tags to better identify the parts of a Web page
    - Canvas tag, which creates a bitmapped work surface
  - No proprietary software is required
Example of an HTML5 Web page

FIGURE 11-27
An example of HTML5 source code and its corresponding Web page.
Scripting Languages

• Scripting languages
  – Add dynamic content to a Web page
  – Scripts are embedded into a Web page’s code
  – Scripts are used as middleware to tie a Web site to a database
  – Scripts can be run:
    • By the Web browser being used to view the Web page (client-side script)
    • By the Web server hosting the Web page being viewed (server-side script)
Examples of Scripting Languages

• **JavaScript** was developed to enable Web authors to implement interactive content on Web sites
  – Client-side scripts; embedded into Web page source code either as direct commands or a link to an external .js file
• **VBScript** (Visual Basic Scripting Edition) was developed by Microsoft for purposes similar to JavaScript
  – Can be used for both client-side and server-side scripts
• **PHP** is commonly used to create dynamic Web pages
  – Server-side scripts
• **Perl** is one of the most popular languages for writing CGI scripts that accept data from and return data to a Web server
Example of JavaScript

JavaScript opens more information when an item on the navigation bar is pointed to, as well as changes the color of that item.

FIGURE 11-28
JavaScript is commonly used on Web pages.
Quick Quiz (3)

1. An example of a high-level programming language is __________.
   a. Java
   b. assembly language
   c. HTML

2. True or False: Dart programs can only be executed in Dartium, a Dart-enabled browser.

3. The scripting language designed to replace both HTML and XHTML is __________.

Answers:
1) a; 2) False; 3) HTML5
Summary

- Approaches to Program Design and Development
- The Program Development Life Cycle (PDLC)
- Tools for Facilitating Program Development
- Languages Used for Application Development